

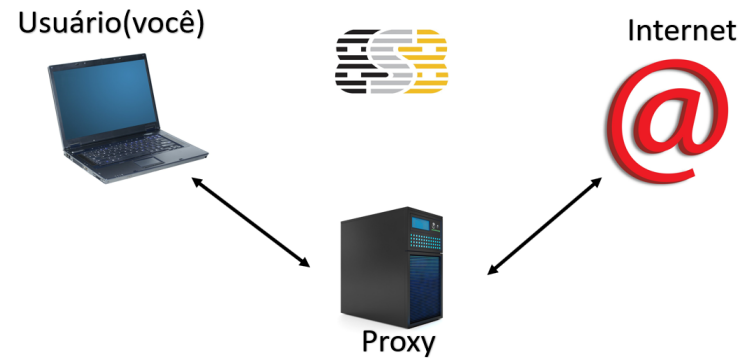
HORA DO BACK-END

Aula -1

-FERNANDO LUCAS (MASSAN)

INICIANDO O SERVIDOR.

Em informática, um servidor é um software ou computador, com sistema de computação centralizada que fornece serviços a uma rede de computadores, chamada de cliente. Esses serviços podem ser de naturezas distintas, como por exemplo, arquivos e correio eletrônico.



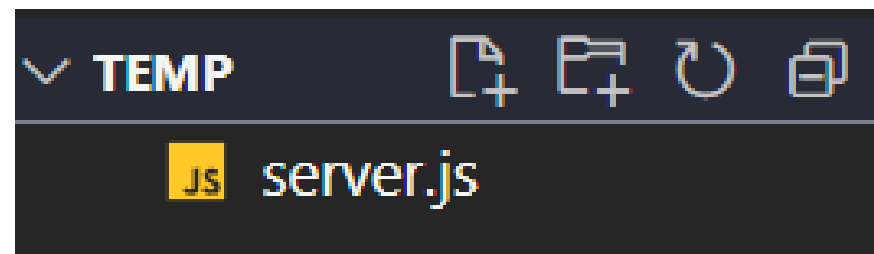
CRIANDO NOVO PROJETO



Vamos criar um novo projeto para iniciarmos nossos estudos de back-end.



Dentro dessa pasta vamos criar nosso primeiro arquivo.



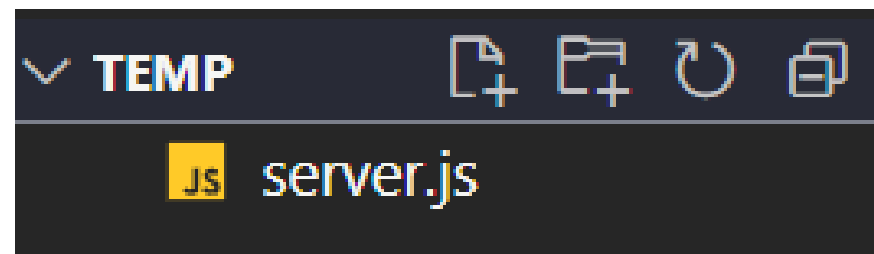
CRIANDO NOVO PROJETO



Vamos criar um novo projeto para iniciarmos nossos estudos de back-end.



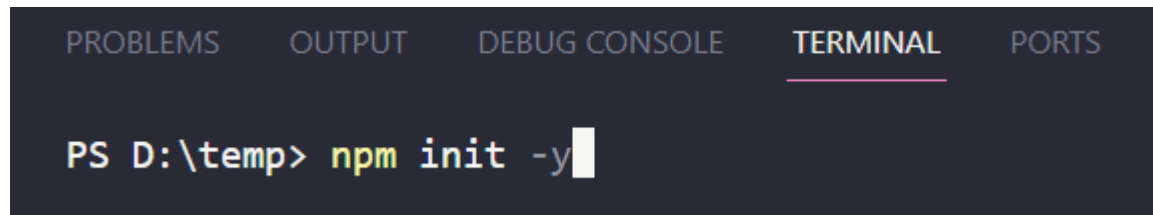
Dentro dessa pasta vamos criar nosso primeiro arquivo.



INSTALANDO DEPENDÊNCIAS

npm é o Gerenciador de Pacotes do Node (Node Package Manager) que vem junto com ele e que é muito útil no desenvolvimento Node. Por anos, o Node tem sido amplamente usado por desenvolvedores JavaScript para compartilhar ferramentas, instalar vários módulos e gerenciar suas dependências.



Abra o terminal e digite o seguinte comando!

A screenshot of a terminal window from a code editor. The terminal has a dark background with light-colored text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. Below the tabs, the prompt 'PS D:\temp>' is followed by the command 'npm init -y' and a white cursor block.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS D:\temp> npm init -y
```

ENTENDENDO O package.json

JSON é uma notação que permite estruturar dados em formato texto para serem utilizados em diferentes tipos de sistemas. Trata-se de um formato simples e leve, que oferece uma série de benefícios, como maior velocidade no tráfego em rede e mais agilidade no processamento.

 package.json
 server.js

```
package.json > ...  
1  {  
2    "name": "temp",  
3    "version": "1.0.0",  
4    "description": "",  
5    "main": "server.js",  
6    "scripts": {  
7      "test": "echo \"Error: no test specified\" && exit 1",  
8      "start": "node server.js"  
9    },  
10   "keywords": [],  
11   "author": "",  
12   "license": "ISC"  
13 }  
14
```

VAMOS INSTALAR O TEMPLATE EXPRESS

Execute o comando `npm install express` note que agora nos nosso arquivo  `package.json`

Temos uma nova dependência o

```
"dependencies": {  
  "express": "^4.17.1"  
}
```

. Repare que também temos uma pasta nova: `> node_modules`. Dentro dessa pasta temos várias dependências que o express precisa para funcionar. Você pode interpretar cada dependência dessa como um programa.

ATENÇÃO



A partir de agora vamos utilizar o JavaScript puro, se você sentir muita dificuldade você deve voltar até o slide "iniciando Javascript" para rever os conteúdos iniciais.

iniciando javascript

Conhecendo o Javascript
Comentários, Strings e Numbers,
Fazendo cálculos com JavaScript,
operadores relacionais e comparativos
Condicionais,
objetos e
Arays.



PowerShell



que a força esteja com você.
Prof. fernando Lucas

NO NOSSO ARQUIVO server.js

Vamos iniciar os comandos em JavaScript

```
const express = require('express')  
const server = express()  
  
server.listen(5000, function() {  
  console.log("server is running")  
})
```

```
npm start
```

CRIANDO ROTAS

Se você acessar servidor no seu navegador "localhost:5000" você perceberá que ele pedirá uma rota. `Cannot GET /`

A rota inicial de qualquer servidor é a "/", então mesmo que a gente não coloque ela no código ela sempre será o primeiro lugar que o código irá procurar.

CRIANDO ROTAS

Já que ele não está encontrando a rota para "/", precisamos indicar pra ele onde encontrar esse lugar.

```
server.get("/", function(req,res){})
```

Vamos utilizar uma função com parâmetros "req" e "res".

Vamos entender essa linha de código.

CRIANDO ROTAS

`server.get` Servidor pegue algo.

`server.get("/",` Servidor pegue a barra

`server.get("/", function` servidor pegue essa barra e execute essa função.

`(req, res)` req = requisição (servidor escuta algo) | res= resposta (servidor responde algo).

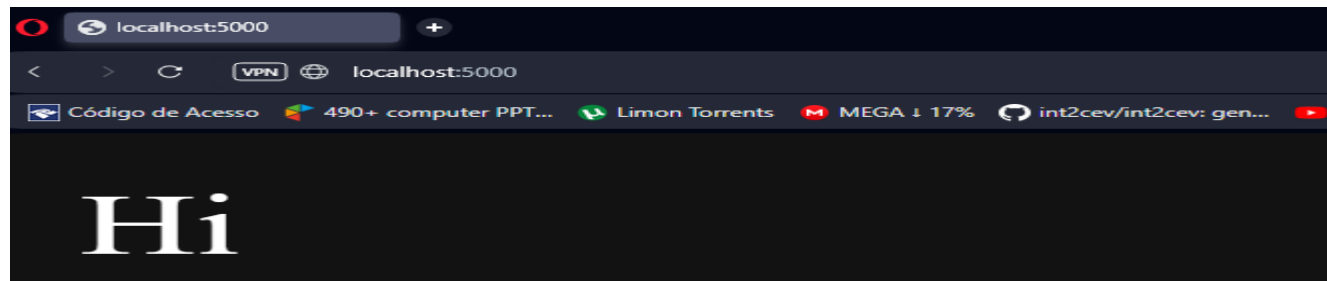
`server.get("/", function(req, res)`

Servidor pegue essa barra e execute a seguinte função, se o usuário enviar algo guarde no parâmetro "req",
Se você for responder algo guarde a resposta no parâmetros "res".

CRIANDO ROTAS

Vamos agora ordenar que o servidor responda a rota "/".

```
server.get("/", function(req,res){  
  ⚡ return res.send("Hi");  
})
```




INSTALANDO DEPENDÊNCIAS

Vamos precisar reiniciar o servidor várias vezes agora.

Para não precisarmos reiniciar toda hora manualmente vamos instalar uma extensão que vai fazer isso de forma automática.

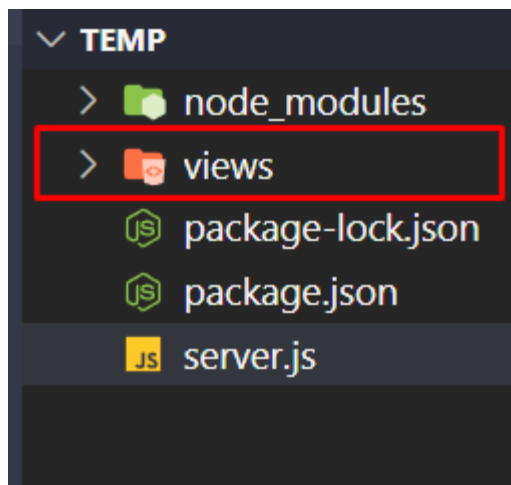
```
npm install -D nodemon
```

```
{  
  "name": "temp",  
  "version": "1.0.0",  
  "description": "",  
  "main": "server.js",  
   Debug  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1",  
    "start": "nodemon server.js"  
  },  
}
```

IMPORTANDO ARQUIVOS

Agora vamos importar nosso site para dentro do nosso servidor.

Crie uma nova pasta chamada de views e arraste seu conteúdo para dentro dela:



```
npm install nunjucks
```

Agora vamos instalar outra extensão para o nosso servidor o nunjucks

O nunjucks é uma template engine que é um pacote de funcionalidades novas para nosso server.

CHAMANDO O NUNJUCKS

```
const express = require ('express');  
const server = express ();  
const nunjucks = require ('nunjucks');  
  
server.set ("view engine","html")  
nunjucks.configure("views",{  
  express:server  
})
```

INICIANDO O SITE

```
server.get("/", function(req, res){  
  return res.render("index");  
});
```

-> dessa forma nos agora vamos renderizar nosso arquivo

HTML no caminho "/".

Teste

Como você pode perceber nosso arquivo css não respondeu, isso se por conta que não instanciamos ele ainda.

INICIANDO O CSS DO SITE

Primeiro vamos criar a linha de código `server.use(express.static('public'))`

Ele vai nos permitir utilizar arquivos estáticos que estejam dentro da pasta public.

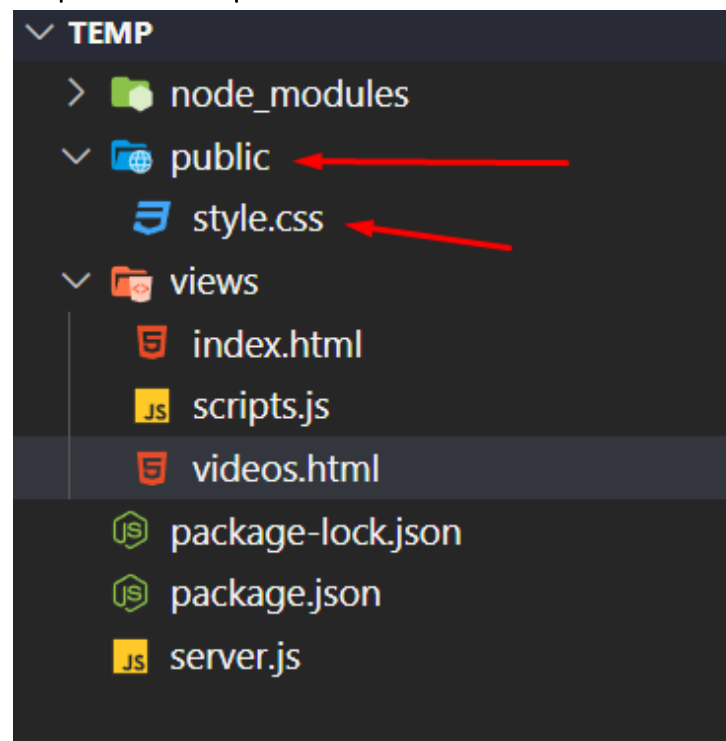
Que é uma pasta que vamos criar e arrastar nosso arquivo CSS pra dentro dela - Conceito public.

Vamos também criar a rota para nossa página de vídeos.

```
server.get("/videos", function(req, res){  
  return res.render("videos");  
})
```

INICIANDO O CSS DO SITE

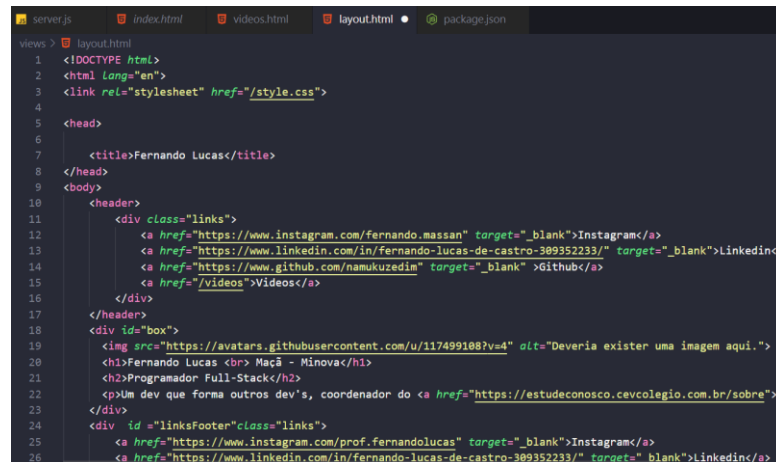
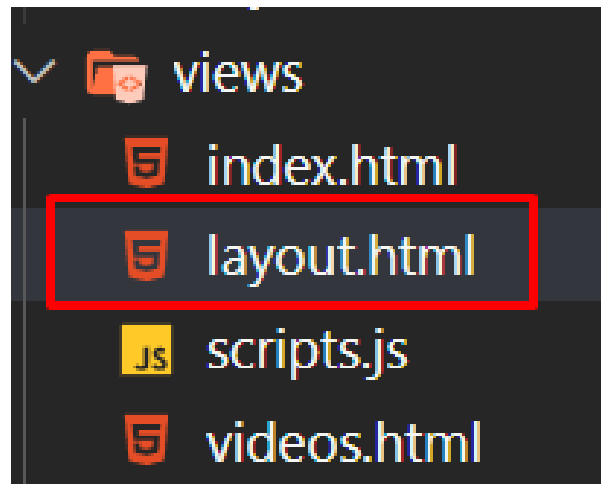
Crie a pasta public e coloque o arquivo css dentro dela..



APROVEITANDO DADOS

Primeiro crie um novo arquivo chamado layout.HTML.

depois copie todo os códigos do index.html para o arquivo layout.htm. Lembre-se de alterar o caminho no servidor.



IMPORTÂNCIA DA INDENTAÇÃO DO CÓDIGO

Agora nós vamos utilizar códigos que já foram processados e renderizados, dessa forma vamos acelerar a velocidade com que conteúdos novos são mostrados.

agora você vai entender visualmente o motivo de
mantermos o código organizado e padronizado para
todas as páginas.

INICIANDO O REAPROVEITAMENTO

views > layout.html > html > body


```
1  <!DOCTYPE html>
2  <html lang="en">
3  <link rel="stylesheet" href="/style.css">
4
5  <head>
6
7      <title>Fernando Lucas</title>
8  </head>
9  <body>
10     <header>
11         <div class="links">
12             <a href="https://www.instagram.com/fernando.massan" target="_blank">Instagram</a>
13             <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
14             <a href="https://www.github.com/namukuzedim" target="_blank">Github</a>
15             <a href="/videos">Videos</a>
16         </div>
17     </header>
18
19     {% block content %}
20
21
22     {% endblock %}
23
24 </body>
25 </html>
```

Va até o arquivo layout e apague todo o conteúdo que não se repete.

Depois adicione os comando {%%}.

INICIANDO O REAPROVEITAMENTO

Agora no arquivo

 *index.html*

Vamos trocar o nome dele para about, e adicionar o comando `{%extends "layout.html"%}`, o arquivo deve ficar assim.

```
{% extends "layout.html" %}
{% block content %}

    <div id="box">
        
        <h1>Fernando Lucas <br> Maçã - Minova</h1>
        <h2>Programador Full-Stack</h2>
        <p>Um dev que forma outros dev's, coordenador do <a href="https://estudeconosco.cevcolegio.com.br/sobre">
    </div>
    <div id="linksFooter" class="links">
        <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram</a>
        <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
        <a href="https://www.github.com/namukuzedim" target="_blank">Github</a>
    </div>
</body>

{% endblock %}
```

Aproveitando dados

DESAFIO -

Use a lógica da funcionalidade `{%extends%}` na página de  `videos.html` ✕

```

views > videos.html > ...
1  {% extends "layout.html" %}
2
3  {% block title %}
4  <title>Fernando Lucas - videos dinamicos </title>
5  <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Ou
6
7  {% endblock %}
8
9
10 {% block content %}
11
12
13
14 > <section class="cards">...
53 </section>
54
55 > <div class="modal_overlay">...
67 </div>
68
69 </div>
70
71 <div id="linksFooter" class="links">
72 <a href="https://www.instagram.com/prof.fernandolucas" target="_blank">Instagram<
73 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target=
74 <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
75
76 </div>
77
78 <script src="scripts.js"></script>
79 {%endblock%}

```

```

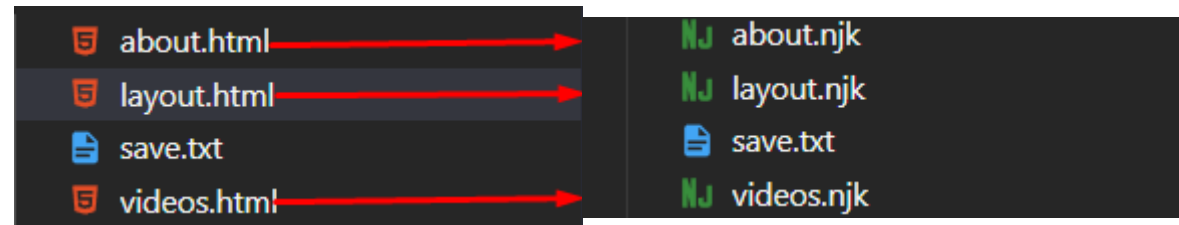
views > layout.html > html > title
1 <!DOCTYPE html>
2 <html lang="en">
3 <link rel="stylesheet" href="/style.css">
4
5  {% block title %}
6  <title>Fernando Lucas </title>
7  {% endblock %}
8
9 <head>
10
11 </head>
12 <body>
13 <header>
14 <div class="links">
15 <a href="https://www.instagram.com/fernando.massan" target="_blank">Instagram</a>
16 <a href="https://www.linkedin.com/in/fernando-lucas-de-castro-309352233/" target="_blank">Linkedin</a>
17 <a href="https://www.github.com/namukuzedim" target="_blank" >Github</a>
18 <a href="/videos">Videos</a>
19 </div>
20 </header>
21
22 {% block content %}
23
24
25
26 {%endblock%}
27
28
29 </body>
30 </html>

```

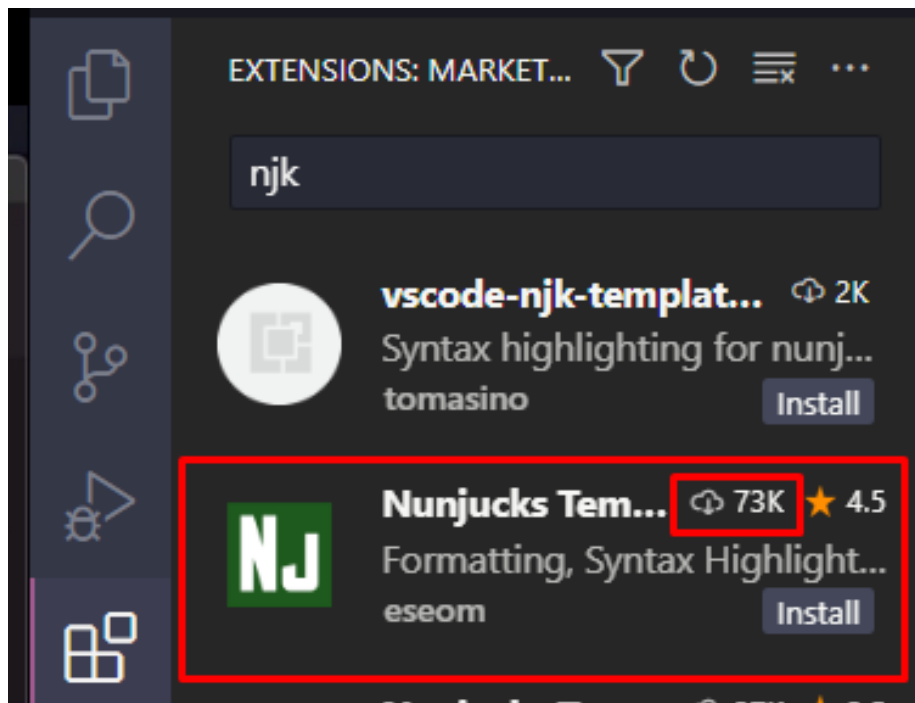
AGORA VAMOS SERVIR OS ARQUIVOS .NJK

Agora nós vamos transformar nossos arquivos HTML em NJK.

```
server.js
server.js > ...
1  const express = require('express');
2  const server = express();
3  const nunjucks = require('nunjucks');
4
5  server.use(express.static('public'))
6
7  server.set(["view engine", "njk"])
8  nunjucks.configure("views", {
9    express: server
10 })
11
12
13 server.get("/", function(req, res) {
14   return res.render("about");
15 })
16 server.get("/videos", function(req, res) {
17   return res.render("videos");
18 })
19
20 server.get("/")
21 server.listen(5000, function() {
22   console.log("server is running")
23 })
```



INSTALE UMA NOVA EXTENSÃO



AGORA REINICIE O VSCODE

VOLTANDO A COMPLETAR HTML

Você deve ter percebido que depois de instalar o a extensão para leitura do .njk, nossos arquivos html (njk) não estão se comportando como antes.

Isso acontece porque precisamos adicionar a configuração do HTML no Nunjucks.

VOLTANDO A COMPLETAR HTML

Nunjucks Template v0.5.1
eseom | 73,917 | ★★★★★ (4)
Formatting, Syntax Highlighting, Hover, and Snippets
Disable | Uninstall | Settings
This extension is enabled globally.

emmet.includeLanguages: {
 "njk": "html"
},

- For vscode embedded emmet, notify that `njk` is html file type

html.format.wrapLineLength: 120

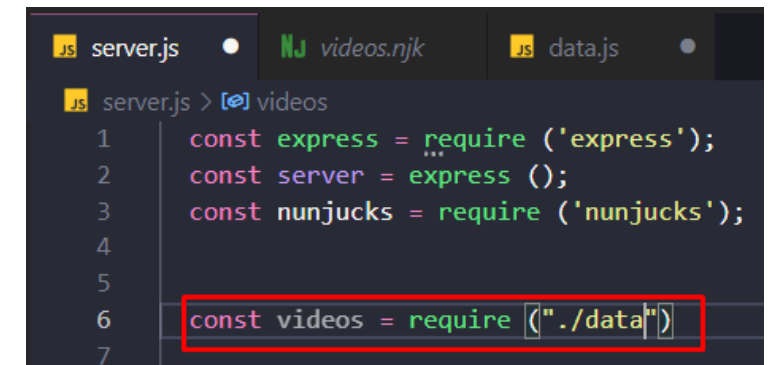
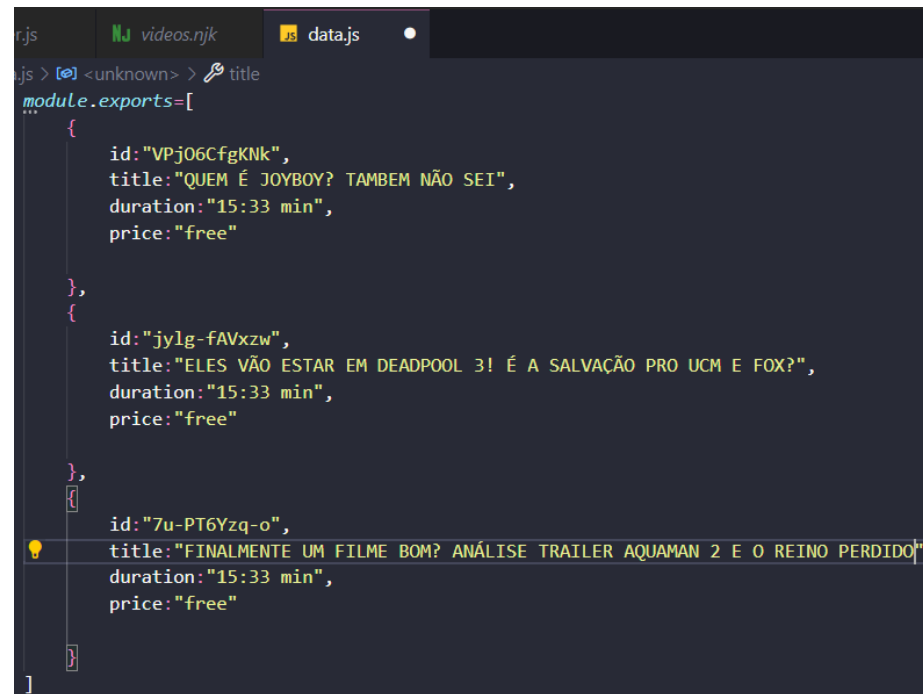
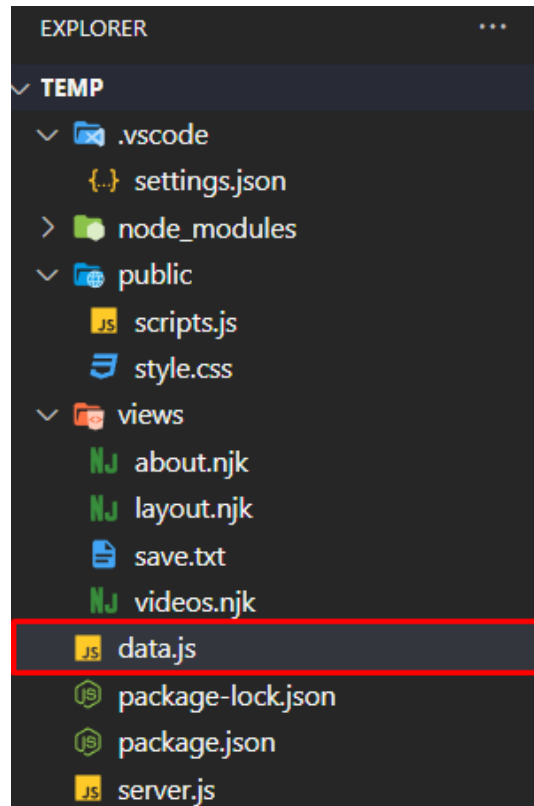
- for vscode-icons ([issue #6](#))

CTRL+SHIFT+P

>settings
Developer: Generate Color Theme From Current Settings

```
.vscode > settings.json > ...  
1 {  
2   "liveServer.settings.port": 5501,  
3   "emmet.includeLanguages": {  
4     "njk": "html"  
5   },  
6 }
```

EXPORTANDO E IMPORTANDO JAVASCRIPT



PASSANDO DADOS DO BACK PARA O FRONT

```
server.js  videos.njk  data.js
server.js > server.get("/videos") callback > items
1  const express = require('express');
2  const server = express();
3  const nunjucks = require('nunjucks');
4
5  const videos = require('./data');
6
7
8
9  server.use(express.static('public'))
10
11  server.set("view engine", "njk")
12  nunjucks.configure("views", {
13    express: server
14  })
15
16
17  server.get("/", function(req, res) {
18    return res.render("about");
19  })
20
21  server.get("/videos", function(req, res) {
22    return res.render("videos", {items: videos});
23  })
```

Apague todos os
cards. ->

```
save.txt  videos.njk  data.js
videos.njk
endblock %}

block content %}

{% for item in items %}
  <section class="cards">
    <div class="card" id="{{item.id}}">
      <div class="card_image">
        
      </div>
      <div class="card_content">
        <p>{{item.title}} </p>
      </div>
      <div class="card_info">
        <p>{{item.duration}}</p>
        <p class="card_price"> {{item.price}} </p>
      </div>
    </div>
  {% endfor %}
</section>
```

DESAFIO -

Entenda como o css não está sendo respeitado no arquivo da página de vídeos e concerte.